Induction Variables and Strength Reduction

- Overview of Optimization
- Algorithms to find induction variables
- Summary: 14.1

A Strength Reduction Example

```
FOR i = 0 to 100
    A[i] = 0;
```

```
    i = 0                                  t2'   = 4 * 0  + &A = &A
                                           t3    = 4*100 + &A
L2: if (i >= 100) goto L1              L2: if (t2' >= t3)
    t1   = 4 * i
    t2   = &A + t1      ==>
    *t2 = 0                                   *t2' = 0
    i    = i + 1
    goto L2
L1:                                            t2' = t2' + 4
                                               goto L2
                                           L1:
```

Definitions

- A **basic induction variable (BIV)** is a variable X
  - whose only definitions within the loop are assignments of the form X = X + c or X = X – c, where c is either a constant or a loop-invariant variable
- An induction variable is
  - a basic induction variable
  - a variable defined once within the loop, whose value is a linear function of some basic induction variable at the time of definition

    A = c1 × B + c2
- The family of a basic induction variable B
  - the set of induction variables A such that each time A is assigned in the loop, the value of A is a linear function of B

Strength Reduction

- Strength Reduction

  Let A be an induction variable in a family of basic induction variable B

  (i.e., $A = c1 \times B + c2$)

  - Create a new variable: A’
  - Initialization in preheader: $A' = c1 \times B + c2$
  - Track value of B: add after $B = B + x$: $A' = A' + x \times c1$
  - Replace assignment to A: $A = A'$

A Strength Reduction Example

- There are two induction variables in the example below
  - t1 = 4 * i (create t1'= 4*i)
  - t2 = 4 * i + &A (create t2'= 4*i + &A)

```
    i = 0
L2: if (i >= 100) goto L1
    t1  = 4 * i
    t2  = &A + t1
    *t2 = 0
    i   = i + 1
    goto L2
L1:
```

Strength Reduction Example

```
FOR i = 0 to 100
    A[i] = 0;
```

```
    i = 0

L2: if (i >= 100) goto L1
    t1   = 4 * i
    t2   = &A + t1      ==>
    *t2  = 0
    i    = i + 1
    goto L2
L1:
```

```
    i    = 0            /* Deleted */
    t1'  = 4 * i = 0    /* Deleted */
    t2'  = 4 * i + &A = &A
    t3   = 4*100 + &A
L2: if (t2' >= t3)
    t1   = t1'          /* Deleted (dead)*/
    t2   = t2'          /* Propagated into */
    *t2  = 0            /* *t2' = 0 */
    i    = i + 1        /* Deleted (dead) */
    t1'  = t1' + 4      /* Deleted (dead) */
    t2'  = t2' + 4
    goto L2
L1:
```

Follow-up Optimizations

- Optimizing non-basic induction variables
  - Copy propagation (e.g., t2)
  - Dead code elimination (e.g., t1)
- Optimizing basic induction variables (e.g., i) eliminates BIVs used only for calculating other induction variables and loop tests
  - Select an induction variable A in the family of B, preferably with simple constants (e.g., A = c1 × B + c2)
  - Replace a comparison such as “if B > x goto L1” by “if (A’ > c1 × x + c2) goto L1”, assuming c1 is positive
    - Called linear function test replacement (LFTR)
  - If B is live at any exit from the loop, recompute it from A’ after the exit, B = (A’ – c2)/c1

Strength Reduction with non-trivial BIVs

- When there are multiple definitions for a BIV

Example:

```
k = 0;
for (i = 0; i < n; i++) {
        k = k + 3;
        . .= m;
        if (x < y)
            k = k + 4;
        if (a < b)
            m = 2 * k;
        k = k – 2;
        . .= m;
}
```

Each iteration may execute a different number of increments/decrements

Strength Reduction with non-trivial BIVs

We can still do strength reduction, removing multiplication

```
k = 0;
 m' = 2 * k = 0;
for (i = 0; i < n; i++) {
            k = k + 3;
              m' = m' + 6;
            . .= m;
            if (x < y) {
                k = k + 4;
                  m' = m' + 8;
                }

            if (a < b)
                m = m';
            k = k – 2;
              m' = m' – 4;
             . .= m;
}
```